ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOBC2

HIGH-PERFORMANCE SIMULATIONS OF COHERENT SYNCHROTRON RADIATION ON MULTICORE GPU AND CPU PLATFORMS *

B. Terzić†, A. Godunov, K. Arumugam, D. Ranjan, M. Zubair
Old Dominion University, Norfolk, VA 23529, USA

Abstract

Coherent synchrotron radiation (CSR) is an effect of self-interaction of an electron bunch as it traverses a curved path. It can cause a significant emittance degradation and microbunching. We present a new high-performance 2D, particle-in-cell code which uses massively parallel multicore GPU/GPU platforms to alleviate computational bottlenecks. The code formulates the CSR problem from first principles by using the retarded scalar and vector potentials to compute the self-interaction fields. The speedup due to the parallel implementation on GPU/CPU platforms exceeds three orders of magnitude, thereby bringing a previously intractable problem within reach. The accuracy of the code is verified against analytic 1D solutions (rigid bunch).

INTRODUCTION

Coherent synchrotron radiation (CSR) can lead to a host of deleterious effects, such as increase in emittance and energy spread and microbunching instability. First step in mitigating these adverse effects is developing a code for high-fidelity numerical simulation of CSR. Numerical simulations of the CSR effects based on Greens function approach have proven to be extremely challenging because of the following features of simulation: (i) large memory requirement associated with storing the history of the beam bunch; (ii) difficulty to accurately account for retardation; (iii) large cancellation between $\boldsymbol{E}$ and $\boldsymbol{B}$ fields in Lorentz force; (iv) sensitivity to numerical noise, exacerbated by presence of gradients in relevant equations; (v) the manner in which self-interactions scale. In this paper, we present a new, 2D particle-in-cell code for modeling CSR and other collective effects in an electron beam using state-of-the-art computing platforms. The proposed method is optimized to run efficiently on different computing platforms such as GPUs, multicore CPUs and on hybrid CPU-GPU. This adaptation of the code design to the new computing architectures results in unprecedented efficiency and fidelity.

EQUATIONS OF MOTION

The dynamics of electron beams is captured by the Lorentz force:

$$\frac{d}{dt}\left(\gamma m_e \boldsymbol{v}\right) = e\left(\boldsymbol{E} + \boldsymbol{\beta}\times\boldsymbol{B}\right), \tag{1}$$

where the relativistic $\boldsymbol{\beta}$ and γ, velocity $\boldsymbol{v}$, electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{B}$ given as, respectively,

$$\boldsymbol{\beta} \equiv \boldsymbol{v}/c,\ \ \gamma = \frac{1}{\sqrt{1+\boldsymbol{\beta}^2}},\ \ \boldsymbol{v}(\boldsymbol{p}) = \frac{\boldsymbol{p}/m_e}{\sqrt{1+\boldsymbol{p}\cdot\boldsymbol{p}/(m_e c)^2}}, \tag{2a}$$

$$\boldsymbol{E} = -\nabla\phi - \frac{1}{c}\frac{\partial \boldsymbol{A}}{\partial t},\qquad \boldsymbol{B} = \nabla\times\boldsymbol{A}. \tag{2b}$$

ϕ and $\boldsymbol{A}$ the *retarded scalar and vector potentials*, respectively, which are computed by integration of the charge distribution ρ and charge current density $\boldsymbol{J}$ over the *retarded time* $t' = t - |\boldsymbol{r}-\boldsymbol{r}'|/c$:

$$\begin{bmatrix}\phi(\boldsymbol{r},t)\\ \boldsymbol{A}(\boldsymbol{r},t)\end{bmatrix} = \int_0^\infty \begin{bmatrix}\rho(\boldsymbol{r}',t-\frac{\boldsymbol{r}-\boldsymbol{r}'}{c})\\ \boldsymbol{J}(\boldsymbol{r}',t-\frac{\boldsymbol{r}-\boldsymbol{r}'}{c})\end{bmatrix}\frac{d^2\boldsymbol{r}'}{|\boldsymbol{r}-\boldsymbol{r}'|}, \tag{3a}$$

$$\begin{bmatrix}\rho(\boldsymbol{r},t)\\ \boldsymbol{J}(\boldsymbol{r},t)\end{bmatrix} = \int_0^\infty \begin{bmatrix}1\\ \boldsymbol{v}(\boldsymbol{p})\end{bmatrix} f(\boldsymbol{r},\boldsymbol{p},t)d\boldsymbol{p}. \tag{3b}$$

$\boldsymbol{r}$ are the particle coordinates, $\boldsymbol{p}$ the particle momentum and $f(\boldsymbol{r},\boldsymbol{p},t)$ is the beam's particle distribution function (DF) in phase space, m_e is electron mass, c the speed of light. It is important to note that $\boldsymbol{E} = \boldsymbol{E}^{\text{ext}}+\boldsymbol{E}^{\text{self}}$, $\boldsymbol{B} = \boldsymbol{B}^{\text{ext}}+\boldsymbol{B}^{\text{self}}$. $\boldsymbol{E}^{\text{ext}}$ and $\boldsymbol{B}^{\text{ext}}$ are external electromagnetic (EM) fields fixed by the accelerator lattice. $\boldsymbol{E}^{\text{self}}$ and $\boldsymbol{B}^{\text{self}}$ are the EM fields from the bunch self-interaction, which depend on the history of the bunch charge distribution ρ and current density $\boldsymbol{J}$ via the retarded scalar and vector potential ϕ and $\boldsymbol{A}$.

As can be seen from Eq. (3a), computation of the retarded potentials requires integration over the history of the charge distribution and current density. It points to the main computational bottlenecks of the CSR simulations: (i) data storage for the time-dependent beam quantities (ρ and $\boldsymbol{J}$); (ii) numerical treatment of retardation and singularity in the integral equation for retarded potentials; and (iii) accurate and efficient multidimensional integration in the equation for retarded potentials.

MODEL

Vlasov-Maxwell equations in CSR simulations can be solved either directly, by sampling the entire phase space of the DF, on a grid or in a appropriate basis [1], or by using a particle tracking approach (*e.g.*, [2, 3]). Computational requirements associated with sampling the entire phase space limit the direct solvers to low dimensions (usually 1D). Tracking methods are less restrictive owing to the fact that the sampling of the phase space is done only through simulation particles. This allows the study in higher-dimensional systems, which gives them a clear advantage and makes them a preferred method for modeling CSR effects [4]. We

* Work supported by the Jefferson Science Associates Project No. 712336 and the U.S. Department of Energy Contract No. DE-AC05-06OR23177 (B.T. and K.A), and of the Modeling and Simulation Graduate Research Fellowship Program by Old Dominion University 2013-2015 (K.A).
† bterzic@odu.edu

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOBC2

use a particle-in-cell (PIC) tracking method [5–7]. PIC codes sample a particle DF with a large number of point-particles, which do not interact directly with each other, but only through a mean-field of the gridded representation.

Calculations in our algorithm are performed in three coordinate frames. Beam dynamics (particle pushing) is done in the *Frenet frame* (FF), (x,s), defined so that $x \equiv r - r_0$ is the horizontal offset from the designed orbit (r_0,θ), and $s \equiv r_0\theta$ is the longitudinal coordinate. Computation of retarded potentials is performed in the *lab frame* (LF), (X,Y), defined as the Cartesian coordinates in the plane of the beam lattice. Grid operations and interpolation is done on the *grid frame* (GF), $(\tilde{X},\tilde{Y})$, which is just is scaled and rotated LF.

Outline of the Algorithm

At the top-most level, the new algorithm consist of four consecutive steps that are computed at each timestep:

1. Deposit the DF sampled by N particles onto a 2D computational grid with resolution (N_X, N_Y) using the PIC deposition scheme described in [5–7], thereby yielding the charge (ρ) and current density $(\boldsymbol{J})$ on each grid point. This involves an inverse interpolation from the particle position to the nearest grid points.
2. Compute retarded potentials on the grid via quadratures defined in Eq. (3a) for all the grid points. This is by far the most computationally-intensive step.
3. Compute the self-forces from Eq. (1) on the grid. Next, for each simulation particle compute the self-forces acting on it by interpolating from the grid. It is required that the particle deposition onto the grid and interpolation from the grid onto particles is done in the same manner, so as to avoid "ghost forces".
4. Use the leap-frog scheme [3] to solve the Lorentz equation in Eq. (1) to advance particles in time by Δt.

The steps 1-4 are repeated until the end of simulations. The coordinates of the rectangular computational 2D grid are first tilted through angle α from the design orbit in the (X,Y) plane, so as to account for the X-Y correlations. The computational grid $G_t = \{\tilde{X}_i, \tilde{Y}_j\}_{j=1,N_Y}^{i=1,N_X}$ at time t is constructed to envelope all particles such that the outliers in the tilted plane are binned into the boundary cells. Orienting the beam in such a way so as to occupy the smallest volume while containing all the particles yields optimal spatial resolution on a fixed-size, rectangular grid. Therefore, at each timestep, the grid is uniquely described by its tilt angle α, physical grid size L_X and L_Y and the location of its center of charge point (X_0, Y_0).

Computing the Retarded Potentials on the Grid

The retarded potentials $\phi(G_{t_k}, t_k)$ and $\boldsymbol{A}(G_{t_k}, t_k)$ for all the grids points on a grid G_{t_k} at time t_k are computed using the quadrature defined in Eq. (3a) which uses general values of $\rho(G_t, t)$ and $\boldsymbol{J}(G_t, t)$ found by interpolation. In order to avoid singularity at $r' = 0$, the integration in Eq. (3a) is performed in polar coordinates:

$$\begin{bmatrix}\phi(\boldsymbol{r},t)\\ \boldsymbol{A}(\boldsymbol{r},t)\end{bmatrix} = \sum_{i=1}^{M_{\text{int}}} \int_0^{R_{\max}} dR' \int_{\theta^i_{\min}}^{\theta^i_{\max}} \begin{bmatrix}\rho(R',\theta',t-\frac{R'}{c})\\ \boldsymbol{J}(R',\theta',t-\frac{R'}{c})\end{bmatrix} d\theta', \tag{4}$$

where M_{int} is the number of "cuts" (up to 4) of the grid by the circle of causality $t' = t - R'/c$. $R_{\max}$ is computed from the circle of causality in Fig. 1.

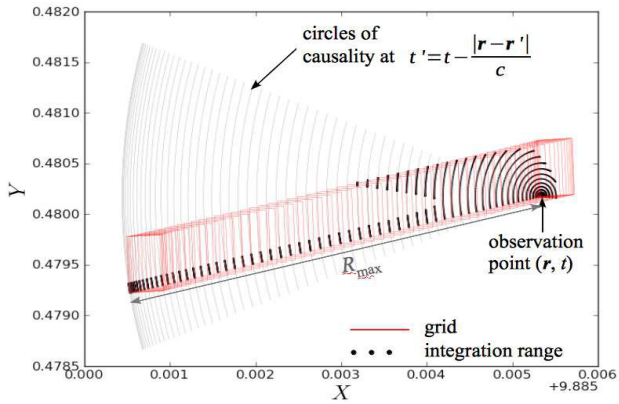

Figure 1: Integration for the retarded potential quadrature in Eq. (4) for a typical grid point. The black lines represent the limits of integration in θ'.

In Eq. (4), the integrand is given at discrete points, and not available in a functional form. Also, the integrand along the outer dimension has regions of high variability as well as regions where change is gradual. In contrast, the inner dimension features only regions where change is gradual. The form of data and the nature of integrand determines the approaches that can be used to evaluate the integral. Newton-Cotes formulas are the most common numerical integration methods used when the integrand is either available in functional form and is "well-behaved" along every dimension or is available only at discrete points. However, when the integrand has regions of high variability then adaptive integration methods are used. Therefore, the nature of integrand in Eq. (4) requires an adaptive integration method to solve the integral along outer dimension and the Newton-Cotes rules along the inner dimension.

Model Validation

We validate our 2D model for simulation of CSR effects in electron beams by comparing our simulation to the only special case for which the exact analytical results are available – that of a 1D monochromatic rigid bunch [8]. Figure 2 shows perfect agreement between the analytic results and our simulation for the LCLS bend [3]. It is noteworthy that each of the curves of the CSR forces represents a difference between two quantities (c.f., Eq. (2b)), which are over 5 orders of magnitude larger. This attests to the accuracy with which the computation of retarded potentials is carried out.

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOBC2

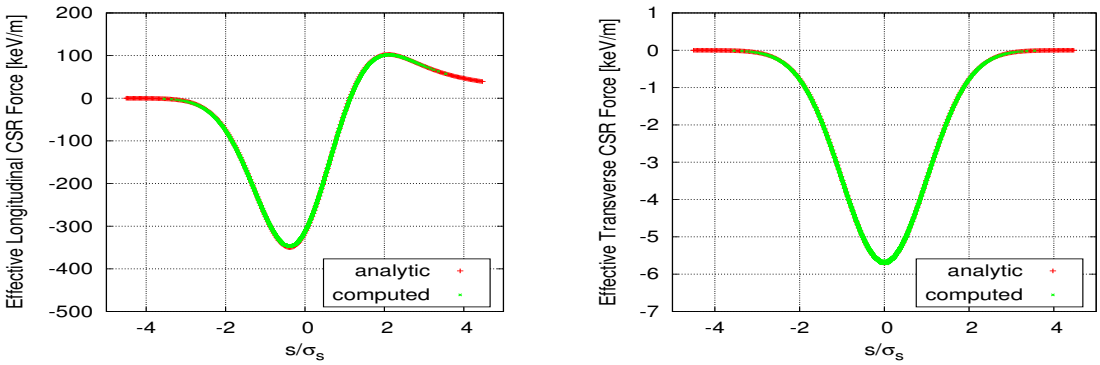

Figure 2: Analytic versus computed effective longitudinal (left) and transverse (right) CSR forces for the LCSL bend [3]: $N = 1024000$ particles on a 64×64 grid, bend radius $R_0 = 25.13$ m, $\theta_b = 11.4°$, longitudinal rms beam size $\sigma_s = 50$ μm, emittance $\epsilon = 1$ nm, and total beam charge of $Q = 1$ nC.

Table 1: Performance results of (a) the multicore CPU implementation running on a standalone desktop machine using one core, (b) the multicore CPU implementation using 8 cores, (c) the GPU implementation using one GTX 480 device, (d) the GPU implementation using 4 GTX 480 device, and (e) the hybrid implementation using all 8 CPU cores and the 4 GTX 480 devices for different sets of input parameters.

Number of Particles (N)	Grid Resolution	Multicore CPU implementation			GPU implementation on a standalone system with				Hybrid implementation on multicore CPU with 4 GPUs	
		Single Core	8 cores		Single GPU		4 GPUs			
		Time (sec.)	Time (sec.)	Speedup	Time (sec.)	Speedup	Time (sec.)	Speedup	Time (sec.)	Speedup
102400	32×32	73.5	11.1	6.6	1.5	49.0	0.7	105.0	0.7	105.0
	64×64	878.5	116.2	7.6	16.8	52.3	4.7	186.9	4.5	195.2
	128×128	13123.2	1695.3	7.7	246.8	53.2	68.4	191.9	65.8	199.4
1024000	32×32	58.1	12.7	4.6	1.2	48.4	0.6	96.8	0.6	96.8
	64×64	573.9	83.9	6.8	11.1	51.7	3.2	179.3	3.1	185.1
	128×128	7651.5	1000.9	7.6	144.1	53.1	40.1	190.8	38.6	198.2
4096000	32×32	57.8	11.9	4.9	1.3	44.5	0.6	96.3	0.6	96.3
	64×64	452.8	66.5	6.8	9.2	49.2	2.4	188.7	2.3	196.8
	128×128	5307.5	725.3	7.3	101.4	52.3	27.1	195.9	26.1	203.4

IMPLEMENTATION ON 3 PLATFORMS: MULTICORE CPU, GPU CLUSTER AND HYBRID CPU/GPU

We developed a scalable two-phase parallel algorithm that uses the multicores of underlying architecture to speed up the computations of CSR simulation. In Phase 1, the algorithm approximates the $N_X N_Y$ different quadratures by adaptively locating the subregions of different quadratures in parallel where the error estimate is greater than some user-specified error tolerance. In Phase 2, it calculates in parallel the integral and error estimates on these subregions [9, 10].

On the Intel® Xeon® processor with 8 cores, the CSR simulation using all the 8 CPU cores is up to 7.7 times faster than the optimized code running on a single core of the CPU, yielding a nearly-linear speedup with the number of cores.

On a single GTX 480 GPU, the CSR simulation achives a speedup of over 50. The number of GPU devices that can be used per node is limited by the hardware capability of the underlying compute node (usually 4, as in our cluster; maximum of 8). For the simulation on 4 GPUs attached to the same node, the speedup over a simulation on a single GPU is nearly linear (up to 3.8). We use the cluster implementation to scale the performance beyond 4 GPU devices. The speedup will be even more substantial for GPU clusters with new, more powerful architectures.

The performance of hybrid CPU/GPU implementation is evaluated on a machine using all 8 CPU cores and 4 GTX 480 GPU devices. The maximum speedup for the hybrid implementation is nearly the same as that of GPU implementation using 4 GPU devices. The performance benefit obtained by using hybrid CPU-GPU implementation is negligible when compared against the GPU implementation.

The detailed comparison of the three implementations is shown in Table 1.

CONCLUSION

We presented an innovative, high-performance, high-fidelity parallel code for simulation of CSR effects in electron beams using state-of-the-art multicore systems (GPUs, multicore CPUs, and hybrid CPU/GPU platform). This pioneering implementation on different multicore system results in a orders-of-magnitude speedup over its serial version, thereby bringing the previously intractable physics within reach for the first time. The parallel algorithm outperforms the optimized sequential simulation and achieves a performance gain of up to 7.7 times and over 50 times on the Intel Xeon E5630 CPU and GTX 480 GPU, respectively. We also scaled the algorithm on a cluster of multicore systems. The performance gain of the cluster implementation scales nearly linearly with the cluster size.

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOBC2

REFERENCES

[1] C. Bohn, J. Delayen, Phys. Rev. E, 50, 1516 (1994).

[2] G. Bassi, J.A. Ellison, K. Heinemann and R. Warnock, Phys. Rev. ST: Accel. Beams, 12, 080704 (2009).

[3] R. Li, Nucl. Instrum. Meth. Phys. Res. A, 429, 310 (1999).

[4] R. Li, R. Legg, B. Terzić, J.J. Bisognano, R.A. Bosh, Proc. 33rd International FEL Conference (2011).

[5] B. Terzić, I. Pogorelov, C. Bohn, Phys. Rev. ST: Accel. Beams, 10, 034201 (2007).

[6] B. Terzić, G. Bassi, Phys. Rev. ST: Accel. Beams, 14, 070701 (2011).

[7] R. W. Hockney, J. W. Eastwood, *Computer Simulations Using Particles*, (London: Institute of Physics Publishing, 1988).

[8] Ya. S. Derbenev, V. D. Shiltsev, SLAC-Pub-7181 (1996).

[9] K. Arumugam, A. Godunov, D. Ranjan, B. Terzić, M. Zubair, International Conference on Parallel Processing (ICPP) (2013a).

[10] K. Arumugam, A. Godunov, D. Ranjan, B. Terzić, M. Zubair, Proc. of High Performance Computing (HiPC) (2013b).